# Guiando a Adaptação de Frameworks através de Hiperdocumentos de Instanciação

**Julio Cezar Zancan** **Roberto Tom Price**
UFRGS – ILEA / Instituto de Informática
Caixa Postal 15064, Porto Alegre, RS, Brazil
e-mail: {jczancan@ilea.ufrgs.br, tomprice@inf.ufrgs.br}

## Resumo

Frameworks são a implementação de uma estrutura de aplicação genérica e reusável, a qual pode ser especializada para aplicações específicas, através da criação de subclasses ou componentes reusáveis. Frameworks possibilitam a reutilização de projeto e código na construção de aplicações. No entanto, são requeridos mecanismos para conduzir e assistir usuários de frameworks na construção de aplicações, uma vez que estes usuários precisarão realizar adaptações em uma estrutura de software que não criaram. O presente trabalho, apresenta uma abordagem fundamentada no uso de hiperdocumentos de instanciação através dos quais são implementados roteiros para adaptação de aplicações. Nesta perspectiva, a construção de uma aplicação é subdivida em um conjunto de etapas guiadas por um hiperdocumento. As etapas correspondem a nodos (nodos-etapa) no hiperdocumento e determinam um conjunto de tarefas de instanciação que devem ser executadas. A criação de subclasses e a implementação de métodos concretos para métodos abstratos definidos, são exemplos de tarefas que podem ser executadas. Através de scripts de instanciação associados a etapas, é possível a pré-programação das tarefas que devem ser executadas. Quando um hiperdocumento de instanciação é navegado, os scripts podem ser ativados por um usuário para guiá-lo na execução de tarefas de instanciação. Durante a execução de scripts, é possível orientar o usuário, ativar ferramentas para execução de tarefas de instanciação (tais como editores e browsers de classe), além de apresentar exemplos de implementação. Utilizando-se de diálogos interativos, scripts podem obter detalhes do usuário e também gerar código da aplicação. Nodos-etapas permitem a agregação de documentação de projeto bem como a ativação de exemplos de aplicações. Estes recursos podem ser utilizados em conjunto a documentação textual, por programadores de scripts, como ferramentas de suporte ao entendimento de frameworks e construção de aplicações. Esta abordagem, hiperdocumentos com nodos de scripts, pode ser utilizada como uma ferramenta assistente genérica no cliclo de vida de muitos tipos de documentos de sistemas baseados em hiperdocumentos, tais como ambientes de desenvolvimento de software, preparação de contratos em ecritórios, etc.

## Abstract

Title: “Driving the adaptation of domain-application-frameworks through hyper-documents of instantiation”

Domain-application-frameworks are the implementation of a generic and reusable structure of an application, which may be specialised in specific applications through the creation of subclasses or the employment of reusable components. Frameworks promote reuse of design and code in the construction of applications. However, mechanisms to guide and assist the users of a framework to build the applications are required, because the user has to perform adaptations on an unknown software structure. This paper presents an approach based on instantiation hyper-documents to construct instantiation tours to adapt applications. With this approach, the building of an application is broken in a set of stages, guided by a hyper-document. The stages correspond to nodes (stage-nodes) in a hyper-document and prescribe a set of instantiation tasks that have to be executed. The creation of subclasses and the implementation of concrete methods for defined abstract methods are examples of required instantiation tasks. The instantiation tasks are pre-programmed in instantiation scripts associated to the stage-nodes. When instantiation hyper-documents are navigated, the scripts associated to the stage-nodes may be activated to help users execute the instantiation tasks. During the execution of scripts, special tools can be activated (such as editors, class-

browser), and examples of code may be shown, to support the user in performing the instantiation tasks. By way of interactive dialogs, scripts can obtain detailed data from the user and generate specific application code. Stage-nodes also support the aggregation of design documentation as well as the running of application examples. These active resources should be used, in conjunction with documentation and instruction texts, by script programmers as tools to aid the user in the tasks of framework understanding and application construction. This approach, of hyper-documents with scripting nodes, may be used as a generic assistant tool to help in the life-cycle process of many types of hyper-document based systems (as software development environments, preparation of contracts in law-offices, etc).

## 1 INTRODUÇÃO

Frameworks orientados a objetos [DEUTSCH 89] [JOHNSON 91] [PREE 94] [FAYAD 97], são uma importante abordagem de reuso para construção de aplicações. Um framework orientado a objetos é um esqueleto de aplicação semi-acabado, constituído de classes abstratas e concretas inter-relacionadas. Este esqueleto semi-acabado provê uma infra-estrutura que pode ser reutilizada na construção de diferentes aplicações pertencentes a um determinado domínio de aplicação. Um framework implementa o fluxo de controle principal, o qual determina, de maneira geral, como serão realizados os serviços nas aplicações construídas a partir do framework. Frameworks introduzem uma inversão no foco de reuso. Na implementação de um framework é construído um fluxo de controle de aplicação, e ao usuário[1] cabe criar os detalhas das operações abstratas especificadas no framework. A construção de uma aplicação, é realizada através da criação de subclasses para adaptar o comportamento genérico das classes abstratas do framework ou através de seleção e conexão de componentes prontos de bibliotecas; em tempo de execução, o framework invoca o comportamento acrescentado pelo usuário.

Através de frameworks, é reutilizado inteiramente o projeto abstrato de aplicações [DEUTSCH 89]. Isto significa que é possível reutilizar as soluções adotadas por um projetista, ou seja a divisão de uma aplicação em um conjunto de classes, e as interfaces e protocolos de colaboração definidos para estas classes. O nível de reuso obtido através de frameworks torna possível aumentar a produtividade no desenvolvimento de aplicações [LAJOIE 94].

No entanto, a inversão de controle introduzida por frameworks no desenvolvimento de aplicações, implica, em geral, na necessidade de usuários compreenderem previamente o framework para construírem suas aplicações. Compreender as fatorações de classes adotadas e os complexos protocolos de colaborações estabelecidos no projeto de um framework é uma atividade complexa que consome esforços significativos de usuários.

É necessário prover a usuários, de diferentes tipos de documentação para apoiá-los na construção de aplicações. São necessárias desde instruções de uso, que enumeram os passos que usuários devem seguir para construção de aplicações, até documentação detalhada de projeto, a qual descreve a organização da estrutura de classes e os protocolos de colaborações implementados em um framework. Diversas técnicas de documentação foram propostas para suprir necessidades de documentação na instanciação de frameworks como CookBooks, Motifs [JOHNSON 92], Contratos [HELM 90] e Meta-Padrões [PREE 94]. Porém recursos documentacionais, apenas possibilitam amenizar o trabalho na instanciação de aplicações. O usuário continua sendo inteiramente responsável por conhecer o framework e escrever código de sua aplicação.

Usuários de frameworks necessitam de mecanismos para conduzi-los na construção de aplicações e permitir a integração de recursos de apoio à compreensão e apoio à instanciação. É necessário descrever e conduzir o usuário nos caminhos que devem ser seguidos na instanciação de aplicações. Para a realização das atividades envolvidas na construção de aplicações, é preciso prover o usuário de explicações, documentação e exemplos de implementação. Tais mecanismos poderiam diminuir a dificuldade para o usuário na instanciação de aplicações, permitindo automatizar a execução de atividades de instanciação

[1] Programador que cria aplicações a partir do framework

Este artigo descreve uma ferramenta desenvolvida para apoio à instanciação de frameworks, a qual é baseada na utilização de *hiperdocumentos de instanciação*. Através de hiperdocumentos de instanciação, é possível modelar roteiros de instanciação, os quais determinam etapas e tarefas que devem ser cumpridas na construção de aplicações a partir de frameworks. Um tipo de nodo especial associado a hiperdocumentos de instanciação, são os *scripts de instanciação*. Scripts são um mecanismo para guiar ativamente o usuário na execução de tarefas de instanciação. Durante a execução de um script, são apresentadas instruções ao usuário, diálogos para o fornecimento de características da aplicação, gerados componentes, além de ativadas ferramentas para realização de atividades específicas de instanciação. Paralelamente, hiperdocumentos de instanciação possibilitam a agregação de diferentes tipos de recursos de apoio como documentação diagramática e exemplos de aplicação, que complementam as explicações e servem como subsídio ao usuário na execução de tarefas de instanciação.

## 2 FRAMEWORKS

A estrutura principal de um framework é constituída de um conjunto de classes abstratas [JOHNSON 91]. Uma definição simplificada de classes abstrata, é: uma classe que contém pelo menos um método cuja implementação está indefinida. É delegada às subclasses da classe abstrata, a responsabilidade de fornecer uma implementação para os métodos abstratos especificados na classe abstrata. No entanto uma classe abstrata é uma técnica de projeto, a qual possibilita a construção de componentes de software genéricos, aumentando o potencial de reusabilidade da orientação a objetos. Classes abstratas implementam, além de métodos abstratos, outros três tipos de métodos: métodos *template*, métodos *hook*, e métodos base.

Métodos templates caracterizam-se como algoritmos genéricos, os quais determinam como uma classe abstrata colaborará com outros métodos, para realizar serviços que são de sua responsabilidade. Métodos template invocam métodos abstratos e métodos hook. Esta característica torna possível adaptar o comportamento final de métodos template, criando subclasses para fornecer uma implementação aos métodos abstratos e hooks, invocados pelo método template.

Métodos hook, são outro mecanismo para adaptar o comportamento em classes abstratas. No entanto, ao contrário de métodos abstratos, métodos hook, possuem uma implementação inicial, a qual pode ser redefinida em subclasses da classe abstrata.

Métodos base, são métodos que implementam funcionalidade de uso geral. Tais métodos não são utilizados pela classe abstrata, mas apenas por clientes desta (subclasses ou outras classes que interagem com a classe abstrata). Métodos base, são implementados em classes abstratas por razões de acessibilidade a subclasses.

A figura 1 apresenta um esquema ilustrando o comportamento de métodos de uma classe abstrata. O algoritmo implementado no método template, invoca métodos abstratos e hooks definidos em classes abstratas. Os métodos abstratos e hooks, serão implementados ou redefinidos em subclasses das classes abstratas.

Classes abstratas servem como um molde para construção de subclasses, provendo especificação e projeto. Deutch considera uma classe abstrata já como framework de uma única classe [DEUTSCH 89].

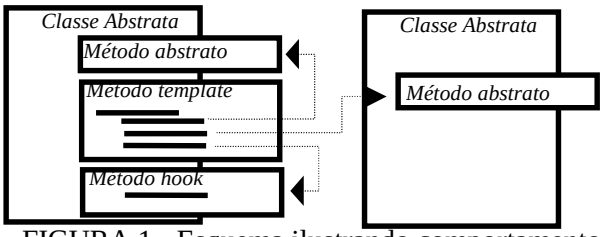

FIGURA 1 - Esquema ilustrando comportamento de métodos de classes abstratas

Uma estrutura de classe composta essencialmente de classes abstratas interelacionadas, constituem um framework orientado a objetos. Ao invés de prover especificação apenas para a construção de único componente, a estrutura de classes de um framework, provê especificação para a construção de uma aplicação completa. Os métodos template da estrutura de classes abstratas determinam genericamente como aplicações construídas com um framework comportar-se-ão. O aspecto de destaque, é que as classes

abstratas de um framework são projetadas para serem reutilizadas em conjunto, uma vez que colaboram entre si, para a execução de serviços do framework.

Uma das formas através da qual são construídas aplicações a partir de frameworks é criar um conjunto de subclasses para implementação de métodos abstratos e hooks, existentes nas classes abstratas. A figura 2, apresenta uma visão estrutural de uma aplicação construída a partir de um framework, através da escrita de subclasses [CAMPO 97]. O nível superior representa a estrutura de classes do framework, constituída essencialmente de classes abstratas que colaboram entre si através de troca de mensagens. O nível inferior, representa subclasses desenvolvidas por um usuário na construção de uma aplicação específica. Na figura 2, também é possível ser observada a inversão de controle introduzida por frameworks. O comportamento implementado nas subclasses de uma aplicação, é invocado em tempo de execução pelos algoritmos de métodos template das classes abstratas do framework.

Outra forma, porém menos freqüente, de construírem-se aplicações a partir de frameworks, é através da seleção de componentes prontos de bibliotecas, os quais podem ser conectados ao framework. Neste caso o usuário implementa subclasses apenas para criar e compor conjuntos de objetos que serão parte de uma aplicação.

A maior vantagem dos frameworks, é a possibilidade de reusar inteiramente o projeto abstrato de aplicações. Ao construir-se uma aplicação a partir de um framework, se está reutilizando as soluções produzidas por um projetista para a fatoração da aplicação em conjunto de classes e as interfaces e protocolos de colaboração definidos sobre estas classes.

### 2.1 O Problema

A inversão de controle introduzida na construção de aplicações a partir de frameworks, implica que um usuário terá que adaptar uma estrutura de software desconhecida. Por esta razão, é necessário que, em geral, o usuário compreenda um framework antes de utilizá-lo na construção de uma aplicação.

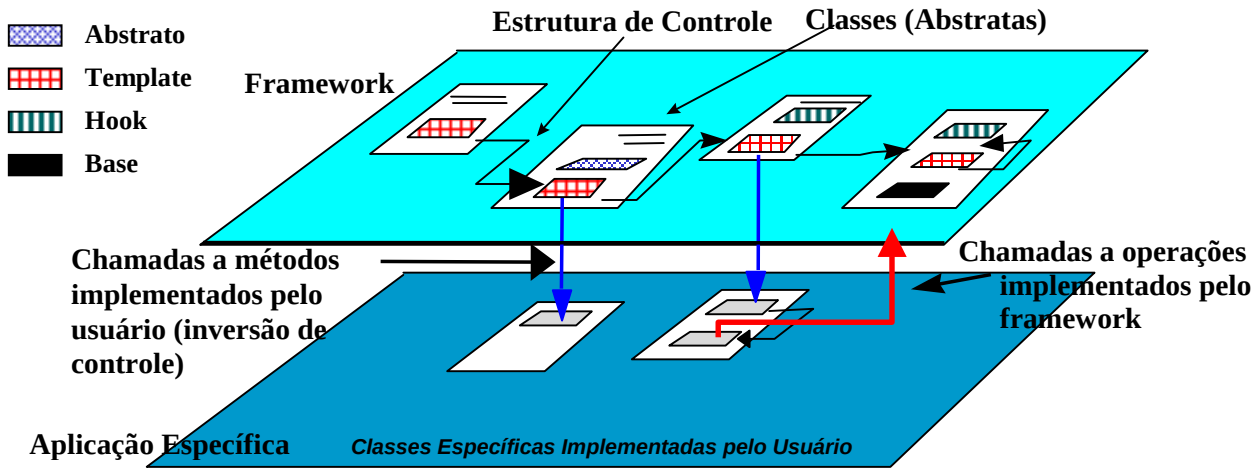


Figura 2 - Visão estrutural de aplicação construída a partir de frameworks [CAMPO 97]

Compreender frameworks é uma tarefa complexa. Frameworks são estruturas de projeto abstratas, nas quais projetistas, com o objetivo de obter o máximo de flexibilidade e generalidade, utilizam-se de soluções elaboradas. Adicionalmente, a própria orientação a objetos impõe dificuldades na compreensão de programas, principalmente para o entendimento do fluxo de controle [CAMPO 97]. A ativação de um método qualquer, inicia uma nova linha de execução (*thread*), na qual são trocadas inúmeras mensagens. Recursos como herança, polimorfismo e acoplamento dinâmico, o quais permitem que o método que será ativado em função do recebecimento de uma mensagem por um objeto, seja definido em tempo de execução, dificulta ainda mais a compreensão de programas orientados a objetos.

### 2.1.2 Abordagens propostas para instanciação de frameworks

Diferentes recursos são necessários para apoiar à instanciação de frameworks. Estes recursos incluem manuais de uso, documentação de projeto, ferramentas de apoio à instanciação além de outros. A instanciação de uma aplicação a partir de um framework implica na execução de diferentes tarefas, as quais demandam conhecimentos distintos do usuário. Exemplos de técnicas de documentação que foram propostas para apoiar à instanciação de frameworks incluem CookBooks, Motifs [JOHNSON 92], Meta-Padrões [PREE 94] e Contratos [HELM 90].

O auxílio provido por cada técnica de documentação, é necessário em algum momento da instanciação de um framework. Cookbooks, por exemplo, fornecem explicações textuais e enumeram passos que usuários devem seguir na instanciação de uma aplicação. Cookbooks são documentação de uso. Já Contratos, utilizam-se de uma linguagem semi-formal, para descrever explicitamente colaborações que ocorrem no interior de um framework. Contratos constituem documentação de projeto. O maior problema é que cada técnica de documentação, é uma proposta isolada. A utilização de uma técnica específica, implica em prover apenas um dos tipos de auxílio necessário à instanciação. Além disto, mesmo com acesso a documentação, o usuário ainda é responsável por compreender o framework e implementar o código de sua aplicação, o que continua sendo uma tarefa trabalhosa.

Uma alternativa para diminuir a dificuldade na utilização de frameworks, é a utilização de ferramentas especializadas como toolkits [JOHNSON 91]. Toolkits para construção de interfaces, por exemplo, permitem que usuários construam visualmente a interface de aplicações, selecionando componentes como listas, caixas de entrada e botões os quais são aplicados em uma janela que representa a interface da aplicação. Após configurar as propriedades dos componentes utilizados, o toolkit gera parte da implementação da aplicação para o usuário.

Ferramentas como os toolkits, contudo, são difíceis de serem construídas e adaptadas. A utilização destas ferramentas é possível somente quando à instanciação pode ser executada através da seleção e composição de componentes prontos de bibliotecas. Porém são poucos os frameworks que podem serem instanciados desta maneira.

## 3 A ABORDAGEM PROPOSTA

A abordagem proposta tem como objetivo principal, conduzir usuários de frameworks na construção de aplicações e integrar os recursos necessários a esta atividade como instruções, documentação e ferramentas. Também pretende-se diminuir o trabalho do usuário na instanciação, através de mecanismos que possibilitem a execução automática de tarefas de instanciação, gerando código da aplicação. Para realizar estes objetivos, são utilizados roteiros de instanciação implementados através de hiperdocumentos.

Roteiros de instanciação tornam possível pré-programar a instanciação de frameworks, estabelecendo etapas e tarefas de instanciação que usuários devem cumprir na construção de aplicações. Hiperdocumentos permitem a implementação de roteiros de instanciação. As etapas de roteiros correspondem a nodos de hiperdocumentos. Através de scripts de instanciação associados a estes nodos-etapas, é possível a programação de tarefas de instanciação.

### 3.1 Roteiros de Instanciação

Para construção de uma aplicação específica a partir de um framework, é necessário cumprir determinadas etapas, nas quais devem ser executadas tarefas de instanciação. As etapas e tarefas que devem ser cumpridas, dependerão das características do framework utilizado e da aplicação que o usuário deseja construir. Um *roteiro de instanciação* é um conjunto específico de etapas, cujas tarefas devem ser cumpridas para a construção de uma determinada aplicação a partir de um dado framework.

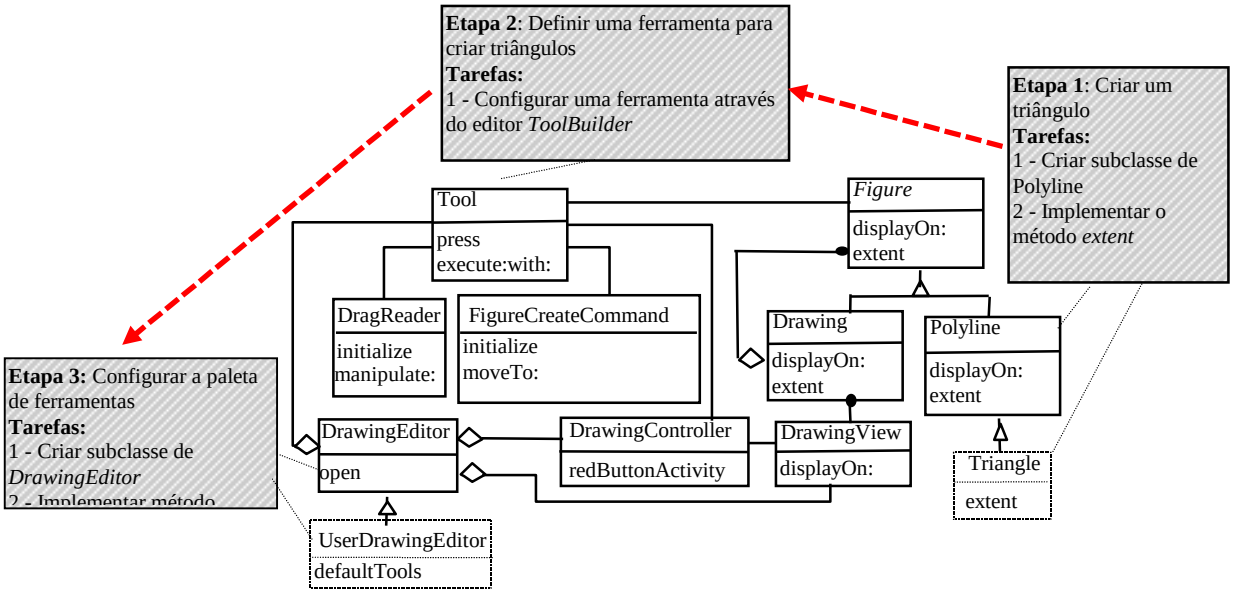


Figura 3 - Exemplo de roteiro de instanciação

A figura 3, apresenta o exemplo de um roteiro de instanciação. As etapas de um roteiro de instanciação estão associadas a determinadas classes de um framework, a partir das quais devem ser executadas tarefas de instanciação. São exemplos de tarefas de instanciação a criação de novas subclasses e a implementação de métodos. Na execução de uma tarefa de instanciação, pode ser utilizada alguma ferramenta especializada. Para executar uma tarefa de instanciação, o usuário precisará de explicações, instruções e documentação de projeto para compreensão do framework.

### 3.2 Roteiros Implementados através de hiperdocumentos

Uma forma para implementar roteiros de instanciação, é a utilização de hiperdocumentos [CONKLIN 87]. Os nodos de um hiperdocumento, correspondem as etapas de um roteiro. Os elos (*links*) entre nodos, correspondem as ligações estabelecidas entre as etapas. Um nodo correspondente a uma etapa contém diferentes explicações sobre tarefas que devem ser executadas. Para guiar o usuário na execução das tarefas de instanciação, é proposta a utilização de nodos ativos, do tipo *scripts de instanciação*. Um script de instanciação é um programa, o qual, quando iniciado, conduz ativamente o usuário na execução de tarefas de instanciação. Durante a execução de um script de instanciação, são descritas tarefas de instanciação, apresentadas instruções, abertos diálogos para a obtenção de características da aplicação, ativadas ferramentas, além de outras atividades. Após a execução de um script de instanciação, é possível que parte da implementação da aplicação tenha sido gerada para o usuário. Scripts de instanciação estão associados a nodos correspondentes a etapas de roteiros de instanciação, os nodos-etapa. Hiperdocumentos utilizados para construção de roteiros e que suportam a utilização de scripts de instanciação, são chamados de *hiperdocumentos de instanciação*.

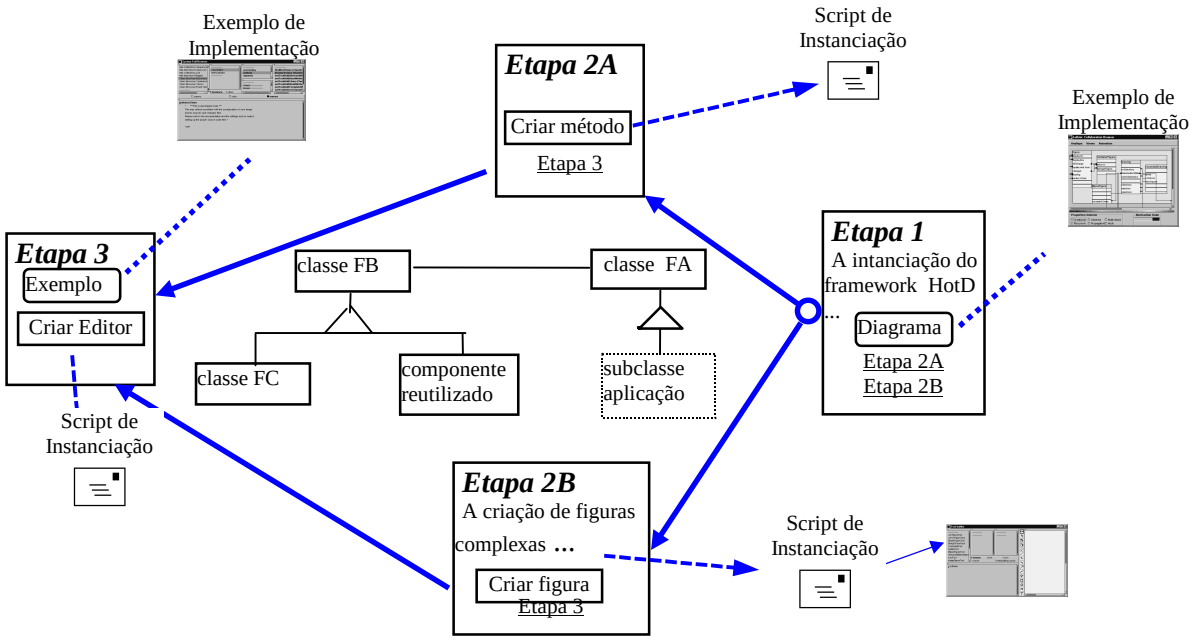


Figura 4 - Exemplo de Hiperdocumento de Instanciação

A figura 4, ilustra um hiperdocumento de instanciação. A estrutura principal de um hiperdocumento de instanciação é constituída por nodos-etapa que representam etapas em roteiros de instanciação, interligados através elos. As interligações entre nodos-etapa, estabelece os *caminhos de navegação* de um hiperdocumento de instanciação. Dependendo dos caminhos seguidos durante a navegação de um hiperdocumento, podem ser produzidas variações nas características finais das aplicações produzidas. Nodos-etapa contém explicações textuais sobre as tarefas que devem ser executados pelo usuário implementador na etapa correspondente. Os textos de nodos etapa podem conter botões que possibilitam a ativação de elos para recursos de apoio à compreensão do framework, os quais incluem documentação diagramática, acesso a exemplo de código fonte e execução de exemplos de aplicações. Além de elos para documentação e exemplos, cada nodo etapa tem associado um script de instanciação.

### 3.2.1 Construção de aplicações através de navegação de hiperdocumentos dirigidos por scripts de instanciação

A construção de aplicações ocorre durante a navegação de um hiperdocumento de instanciação e execução de scripts de instanciação. Para construir uma aplicação através de um hiperdocumento de instanciação, um usuário deve acessar o nodo que representa a etapa inicial e seguir a navegação do hiperdocumento, ativando os scripts de instanciação disponíveis em cada etapa acessada. A medida que o hiperdocumento de instanciação vai sendo navegado e os scripts de instanciação executados, uma aplicação vai sendo construída em conseqüência.

Os nodos etapa do hiperdocumento, apresentam textos explicativos relacionadas ao conjunto de tarefas de instanciação que deverá ser executado em cada estágio da instanciação. Nestes textos, são incluídas explicações sobre aspectos do framework e objetivos de tarefas de instanciação. Também é possível durante o acesso a uma etapa, a ativação de elos existentes junto aos textos, para documentação diagramática, exemplos de código fonte e exemplos de aplicação. Estes recursos servem para complementar explicações textuais contidas nas etapas e como subsídio para a execução de tarefas de instanciação.

As tarefas de instanciação que devem ser executadas em uma determinada etapa, estão pre-estabelecidas em scripts de instanciação. Para executar as tarefas de uma etapa, um usuário deve ativar o script de instanciação associado a etapa. Durante a atividade de um script de instanciação, as tarefas instanciação são apresentadas gradativamente ao usuário, acompanhadas de instruções para sua execução. Parte das tarefas de instanciação podem ser executadas pelo próprio script de instanciação, com participação

do usuário. Um exemplo de tarefa de instanciação realizada desta maneira, é criação de uma nova subclasse. Para a execução de uma tarefa deste tipo, o script de instanciação apresenta um diálogo, no qual o usuário fornece um nome, e em seguida cria automaticamente a subclasse. Também pode ocorrer que durante a execução de um script de instanciação seja ativada uma ferramenta na qual o usuário deve efetuar tarefas. Neste caso, o script de instanciação assiste o usuário, apresentando passos e instruções que devem ser seguidos em cada momento da utilização da ferramenta, para a execução das tarefas de instanciação.

Um hiperdocumento de instanciação apoia à construção de uma aplicação com características específicas a partir de um framework. É possível também a existência de hiperdocumentos de instanciação especializados em fornecer apoio na adaptação de apenas funcionalidades específicas a partir de um framework. Neste casos, aplicações podem ser construídas pela navegação de mais de um hiperdocumento de instanciação.

## 4 FRAMESCRIPT: UM PROTÓTIPO PARA SUPORTE A HIPERDOCUMENTOS DE INSTANCIAÇÃO

Para se testar a viabilidade e eficácia da utilização de hiperdocumentos de instanciação, foi implementado um sistema protótipo de apoio à instanciação [ZANCAN 99]. Este sistema, FrameScript, foi implementado para apoiar à instanciação de frameworks codificados na linguagem Smalltalk-80. Os hiperdocumentos de instanciação, são construídos através de *cartões de instanciação ativos*. O sistema FrameScript dispõe de um conjunto de ferramentas para autoria e navegação de hiperdocumentos de instanciação.

### 4.1 Hiperdocumentos no Sistema FrameScript

Os hiperdocumentos de instanciação utilizados em FrameScript, são construídos baseados no modelo de objetos apresentado na figura 5. Neste modelo, um nodo etapa é representado por um cartão de instanciação ativo. Um cartão de instanciação ativo (classe *Card*) contém textos explicativos (classe *CardText*) e botões (*CardButton*) para acesso a recursos de documentação e exemplos de implementação e aplicações. Cada cartão de instanciação, tem associado um script de instanciação, o qual é representado pela classe *ScriptTemplate*. Os botões para acesso a recursos complementares, também possuem um pequeno script para a ativação editores e browsers, os quais são utilizados na visualização de documentação e exemplos de implementação.

Hiperdocumentos de instanciação, representados pela classe *Hyperdoc*, são uma composição de cartões de instanciação. Cada cartão de instanciação, armazena os elos que possui para os demais cartões de um hiperdocumento de instanciação.

O conteúdo de scripts de instanciação, é constituído de blocos de código implementados em Smalltalk. Scripts de instanciação são interdependentes, de modo que é possível iniciar a implementação de um componente da aplicação em um script de instanciação, e completá-la em outros scripts de instanciação. Para isto ser possível, é necessária a intercomunicação de scripts de instanciação, razão pela qual, scripts possuem parâmetros de entrada e produzem resultados. Hiperdocumentos de instanciação, provém uma memória global, a qual é utilizada para scripts de instanciação armazenarem temporariamente, produtos gerados durante sua atividade, como classes, implementações inacabadas de métodos, nomes, etc. Em scripts que possuem parâmetros, é necessário buscar na memória global valores que serão utilizados nos parâmetros antes de iniciar a sua execução. Esta tarefa é realizada pela configuração da chamada de um script (*ScriptCall*), a qual associa valores da memória global a parâmetros do script, antes de sua execução. É também por intermédio da configuração da chamada, que scripts armazenam resultados na memória global do hiperdocumento.

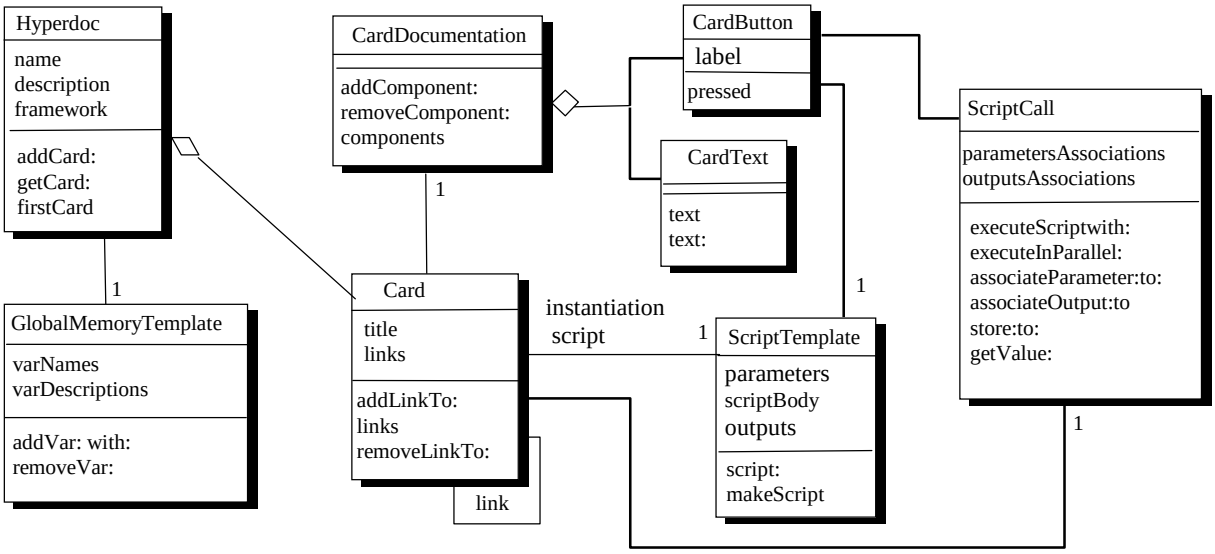


Figura 5 - Modelo de Classes de Hiperdocumentos de Instanciação

### 4.2 Organização do Sistema FrameScript

O sistema é fundamentado na utilização de uma base de hiperdocumentos de instanciação para apoiar a construção de diferentes aplicações a partir de diferentes frameworks. Os hiperdocumentos devem ter sido construídos previamente à instanciação de aplicações, por um *projetista de hiperdocumentos*[2].Diferentes ferramentas foram implementadas, para apoiar a construção e utilização de hiperdocumentos de instanciação, como browser de cartões, browser de hiperdocumentos, editor de scripts.

A figura 6 apresenta a organização geral do sistema FrameScript. Hiperdocumentos de instanciação são armazenados na base de hiperdocumentos do sistema. Um *browser de hiperdocumentos*, possibilita carregar para navegação, hiperdocumentos existentes, bem como iniciar a criação de novos hiperdocumentos. Os hiperdocumentos são navegados através de um *browser de cartões*. O browser de cartões, possibilita visualizar o conteúdo de um cartão de instanciação, ativar botões junto as explicações textuais para acesso a exemplos e documentação diagramática, além de ativar scripts de instanciação e elos existentes entre cartões. O browser de cartões também desempenha funções de autoria, como por exemplo, na definição de elos existentes entre os cartões de instanciação na criação de um hiperdocumento.

As demais ferramentas são exclusivamente de autoria. Através do editor de documentação, é possível definir as explicações que serão incluídas em um cartão, bem como intercalar com as explicações, os botões utilizados para ativação de exemplos e documentação diagramática. O editor de scripts possibilita a construção de scripts em geral. A memória global de hiperdocumentos de instanciação, é definida através do configurador de memória global. O configurador de chamadas de scripts, possibilita definir valores da memória global, que serão utilizados em parâmetros na ativação de um script.

Está sendo utilizada a ferramenta de compreensão de frameworks Meta-Explorer [CAMPO 97], para o fornecimento de documentação diagramatica, como recurso de apoio à compreensão de frameworks. Meta-Explorer permite analisar o comportamento dinâmico de aplicações desenvolvidas a partir de frameworks codificados em Smalltalk, e produzir sofisticados diagramas ilustrando colaborações estabelecidas no framework a partir do qual a aplicação foi construída. Os documentos produzidos a partir de Meta-Explorer podem ser integrados a hiperdocumentos de instanciação, quando a execução de atividades de instanciação, exige uma compreensão mais detalhada do framework.

[2] Neste trabalho, é suposto que o projetista do hiperdocumento seja o próprio projetista do framework ou um usuário especialista nà instanciação do framework.

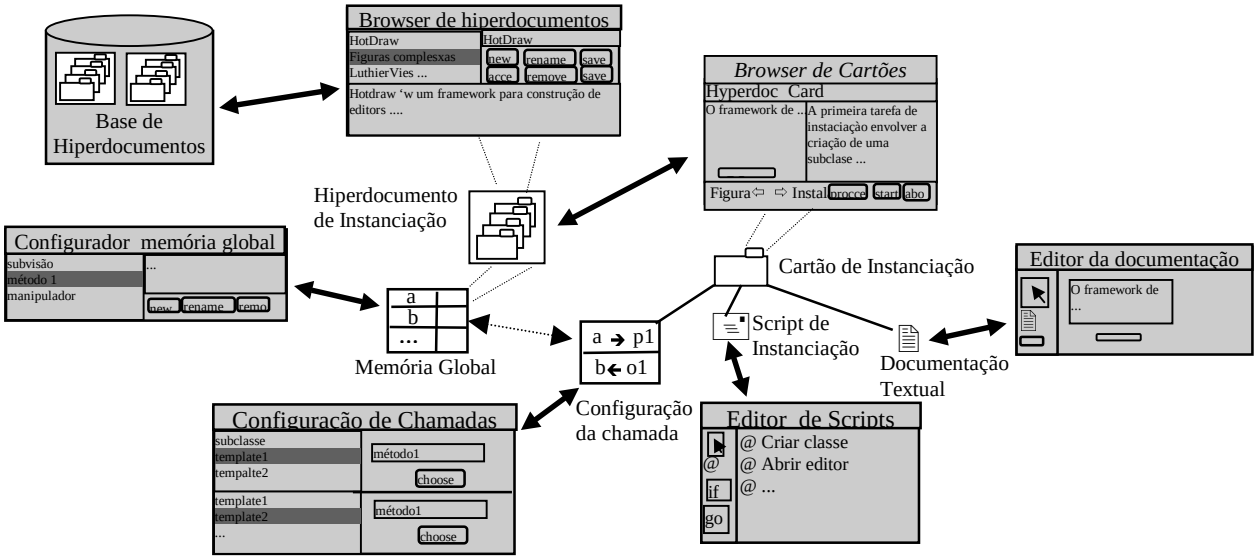


Figura 6 - Organização do Sistema FrameScript

Outros recursos de apoio que estão sendo utilizados, são exemplos de implementação, visualizados através de browsers de classe de Smalltalk, e exemplos de aplicações construídos anteriormente ou existentes no sistema, os quais podem ser rodados pelo usuário durante a navegação de hiperdocumentos de instanciação.

Os recursos citados são integrados a hiperdocumentos de instanciação através de botões mesclados a explicações textuais de cartões de instanciação.

### 4.3 Exemplo de um Script de Instanciação

A figura 7 apresenta o exemplo de um script de instanciação. Este script faz parte de um hiperdocumento para apoio a criação de novos editores gráficos a partir do framework *HotDraw* [JOHNSON 92] [BRANT 94]. Na seção 5, este hiperdocumento será utilizado para exemplificar o uso do Frame-Script.

O script é utilizado para iniciar a criação de um novo editor. A única tarefa realizada no script, é criar uma nova subclasse de DrawingEditor, a qual é a superclasse de editores implementados a partir de HotDraw.

Inicialmente **(1)** é enviada uma mensagem ao usuário, explicando qual o propósito do script e quais tarefas deverão serem realizadas. O identificador *transcript* possibilita acesso a área do browser de cartões, utilizada para enviar mensagens para o usuário. Scripts são interrompidos **(2)** temporariamente para que usuários possam acompanhar as explicações e mensagens produzidas. Quem controla a execução de scripts é o browser de cartões, o qual é representado internamente em scripts pelo identificador *manager*. Para criar uma nova subclasse, é necessário um nome para esta. Em **(3)** é solicitado ao usuário, através de um diálogo, que forneça um nome para o editor. Nesta tarefa é utilizado um componente da biblioteca da linguagem Smalltalk, o qual é *DialogView*. Após fornecido o nome, são realizadas diferentes verificações, como por exemplo, se o nome fornecido é válido e se já não existe uma subclasse com o mesmo nome. Por razões de simplicidade, no script da figura 7 está apenas indicado onde seriam feitas estas verificações.

No próximo trecho , **(4)**, do script é criada a subclasse, a qual é armazenada na variável local *editor*. Para isto é utilizado o componente *CreateSubclass*. Este componente foi desenvolvido no sistema para facilitar o acesso aos serviços de *meta-classes* de Smalltalk [GOLDBERG 89], para a criação de subclasses. Outros componentes deste tipo que foram implementados são *CreateMethod* e *CreateInstVar*. CreateMethod por sua vez, trabalha associado a componentes do tipo *CodeTemplate*, para a criação do corpo de métodos.

```
| nomeEditor editor |

transcript clear; show: 'Através deste script é iniciada a
criação de um novo editor. A tarefa que será realizada consiste
na criação de uma nova subclasse de DrawingEditor.' (1)

manager suspendScript. (2)

nomeEditor := DialogView request: 'Insira um nome para a
subclasse:' withCRs. (3)
        " -- Aqui viriam verificações --"
editor := CreateSubclass on: DrawingEditor with:
nomeEditor in: 'HotDraw-User-Apps'. (4)
manager store: editor to: 'editor'. (5)
transcript cr; show: 'O editor foi criado com sucesso! Script
concluído.' (6).
```

Figura 7 - Exemplo de Script de instanciação

No trecho **(5),** é solicitado ao browser de cartões, para que armazene o editor criado na memória global do hiperdocumento. Isto é necessário, para que outros scripts possam ajudar o usuário a configurar a paleta de ferramentas do novo editor criado. Por fim **(6)**, é apresentada uma mensagem ao usuário, informando que o script foi concluído.

## 5 UM EXEMPLO DE UTILIZACAO DO SISTEMA FRAMESCRIPT

Nesta seção é apresentado um exemplo de utilização do sistema FrameScript, para à instanciação de um editor a partir do framework HotDraw [ZANCA N 99]. HotDraw é um framework para construção de editores gráficos bidimensionais. Um editor construído a partir de HotDraw normalmente possui uma paleta de ferramentas e uma superfície de desenho, na qual figuras podem ser criadas e editadas através de ferramentas selecionadas na paleta. O exemplo descrito a seguir, ilustra a criação de um novo editor e a seleção do conjunto de ferramentas as quais irão compor a paleta de ferramentas do editor. O exemplo será aproveitado também, para apresentação de ferramentas do sistema de apoio à instanciação.


Figura 8 - Browser de Hiperdocumentos

A figura 8 apresenta o browser de hiperdocumentos no qual será • selecionado o hiperdocumento para construção do editor. Através do browser de hiperdocumentos, é possível obter informações sobre o framework o qual aplica-se e o tipo de auxílio oferecido. Na figura 8, está selecionado o hiperdocumento “Criação de Editores”, o qual aplica-se a HotDraw para construção de novos editores.

Hiperdocumentos de instanciaçao são navegados através do browser de hiperdocumentos (figura 9). No painel esquerdo do browser, é apresentada as documentação de um cartão de instanciação que está sendo visualizado, a qual é constituída de textos e botões. O painel direito, é utilizado durante a execução do script do cartão corrente, para serem despejadas instruções e resultados da execução de tarefas de instanciação ao usuário.

Os botões do painel inferior direito, possibilitam controlar a execução de scripts de instanciação. O botão *start* possibilita iniciar o script do cartão corrente. O botão *proceed*, possibilita que o usuário autorize o prosseguimento do script, após serem apresentadas instruções, e o botão abort, possibilita abortar a execução do script corrente. Os elos existentes entre cartões de instanciação, podem ser ativados através no painel inferior esquerdo, logo abaixo da documentação textual do cartão.

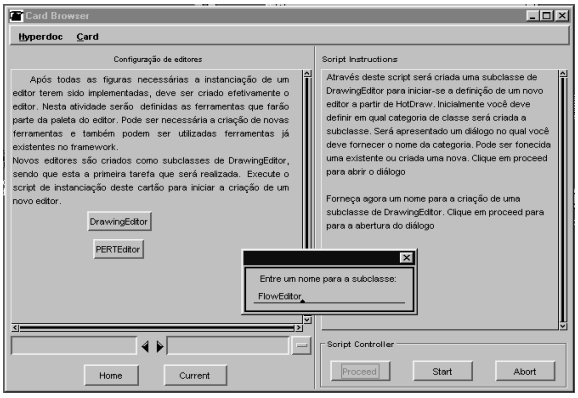

Figura 9 - Cartão para criação de subclasse de DrawingEditor

Retornando ao exemplo, a figura 9 está apresentando o primeiro cartão do hiperdocumento para apoio a construção de editores. Na documentação textual do cartão, é explicado o que é um editor em HotDraw e as tarefas que envolvem a construção de um novo editor. Botões junto à documentação, possibilitam ativar exemplos de editores construídos através de HotDraw. O script deste cartão recebe um nome através de um diálogo e cria uma subclasse, para ser iniciada a configuração de um novo editor (este script foi utilizado como exemplo na figura 7, da seção anterior).

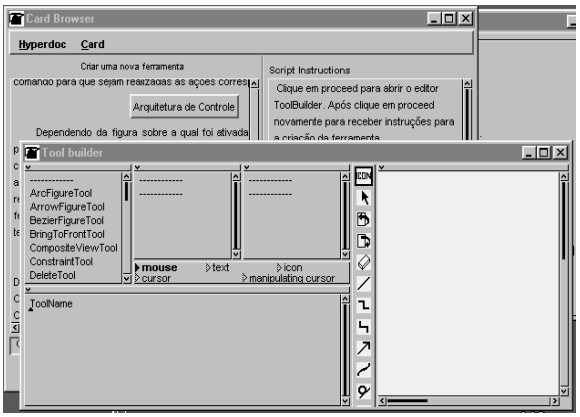

Figura 10 - Cartão para configurar ferramenta

O cartão da figura 10 é utilizado na configuração de novas ferramentas para a criação de novas figuras definidas no framework[3]. O script deste cartão, consiste basicamente em ativar o editor ToolBuilder para criação de ferramentas, e guiar ativamente o usuário na utilização do editor. Um botão junto à documentação deste cartão, possibilita a ativação de um browser de Meta-Explorer, para usuários interessados em entender mais detalhadamente as colaborações na arquitetura de controladores de HotDraw.

O último do cartão do hiperdocumento (figura 11), ajuda o usuário a configurar a paleta de ferramentas do editor. O script de instanciação apresenta uma listagem contendo as ferramentas existentes e as ferramentas definidas pelo usuário. Nesta listagem, o usuário deve selecionar as ferramentas que deseja para seu editor. Após isto, é criado o método *defaultTools* na subclasse, o qual retorna a configuração de ferramentas definida pelo usuário. O botão junto à documentação do cartão, permite visualizar um exemplo de implementação de defaultTools, de outro editor criado com o framework.

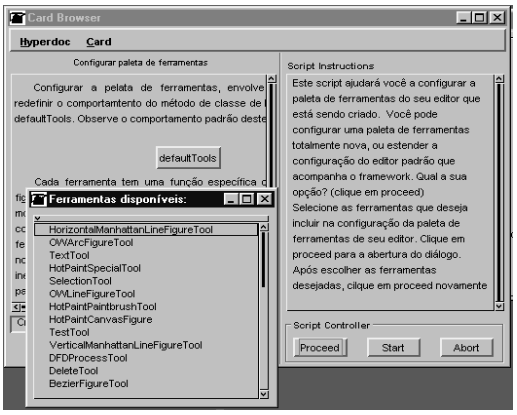

Figura 11 - Cartão para configurar paleta

## 6 CONCLUSÕES

[3] Outro hiperdocumento de instanciação foi desenvolvido no exemplo original, para prover auxílio à criação de figuras em HotDraw.

O principal artefato de reutilização de um framework é o projeto abstrato de uma aplicação, permitindo iniciar a construção de uma aplicação específica em um estágio em que muitas das etapas do ciclo de desenvolvimento já foram realizadas. No entanto são necessários recursos para auxiliar e conduzir usuários de frameworks na instanciação de aplicações. Este artigo apresenta uma proposta para apoiar à instanciação de frameworks através de hiperdocumentos de instanciação. Hiperdocumentos de instanciação possibilitam a programação de roteiros para construção de aplicações. Através de hiperdocumentos é possível descrever etapas e tarefas que devem ser cumpridas na construção de aplicações. Scripts de instanciação, possibilitam conduzir ativamente a execução de tarefas de instanciação e gerar automaticamente código da aplicação em construção. Hiperdocumentos também possibilitam agregar recursos de apoio a utilização de frameworks, como documentação diagramatica e exemplos de aplicação. Um sistema protótipo, FrameScript, foi construído para validar estas idéias em Smalltalk. Neste sistema, foram desenvolvidas ferramentas para apoio ao desenvolvimento de hiperdocumentos de instanciação. Um sistema de apoio à instanciação do framework HotDraw foi desenvolvido como exemplo.

### 6.1. Contribuições

As contribuições da abordagem apresentada, mais extensivamente elaborados na dissertação de mestrado defendida em abril [ZANCAN 99] , podem ser sumarizadas como:

- Utilização do conceito de hiperdocumento para implementar roteiros de instanciação de frameworks

- Sugestão de utilizar scripts de tarefas em nodos-etapa

- Implementação de uma estrutura de nodos-etapa, constituído de textos explicativos, botões de ativação de ferramentas e linguagem de scripting

- Utilizaçâo de Smalltalk para linguagem de scripting.

- Elaboração de uma arquitetura para a execução de scripts utilizando ferramentas existentes como MetaExplorer, Smalltalk browser e editores de frameworks.

- Implantação de ferramentas de apoio a construção de scripts e hiperdocumentos de instanciação

- Implantação de um protótipo de apoio a utilização de hiperdocumentos

- Elaboração de roteiro para framework existente

### 6.2 Limitações

Dentre as limitações da abordagem proposta podem ser citadas:

- A efetividade de uso do sistema de apoio à instanciação, depende da previsibilidade do projetista na construção de hiperdocumentos de instanciação. A obtenção de apoio do sistema para a instanciação de uma determinada aplicação requer que o hiperdocumento apropriado tenha sido construído previamente.

- A situação ideal seria a possibilidade de execução automática total das atividades de instanciação, evitando necessidade de compreensão. No entanto mecanismos de geração de código são rígidos, e em geral não possibilitam produzir todo o código da aplicação.

- Atualmente a construção de hiperdocumentos de instanciação, é trabalhosa e requer projetistas conhecedores de Smalltalk para produção de scripts de instanciação.

### 6.3 Extensões e trabalhos futuros

A forma como hiperdocumentos permitem conduzir usuários na instanciação de frameworks, pode ser generalizada para apoiar à instanciação de outros tipos de documentos. O sistema pode ser extendido para oferecer um mecanismo de modelagem do processos (*workflow*) de adaptação de documentos em geral. Num ambiente de apoio ao desenvolvimento de software, um sistema deste tipo poderia ser utilizado, por exemplo, para projetar hiperdocumentos para conduzir usuários nas diferentes fases do ciclo de desenvolvimento de sistemas. Em cada fase, scripts podem ser utilizados para ajudar da criação dos diferentes documentos envolvidos no desenvolvimento do sistema.

Um dos trabalhos futuros que deverá ser realizado, é uma validação mais ampla do sistema, a partir da construção e utilização de outros modelos de instanciação.

Estudos devem ser realizados para averiguar como os mecanismos utilizados no sistema de apoio à instanciação, poderão ser estendidos para documentos em geral.

Posteriormente, após a generalização dos conceitos e componentes do sistema, poderão ser realizados melhoramentos na interface de criação e utilização de hiperdocumentos. A intenção é tornar mais fácil e visual, principalmente a construção de hiperdocumentos.

**REFERÊNCIAS**


[BRANT 92] BRANT, J. **HotDraw**. [S.l]: University of Illinois, 1992. Master Tesis.

[CAMPO 97] CAMPO, M.; PRICE, R. **Compreensão Visual de Frameworks através da Introspecção de Exemplos**. Porto Alegre: CPGCC da UFRGS, 1997. Tese de doutorado.

[CONKLIN 87] CONKLIN, J. Hypertext: An Introduction and Survey. **IEEE Computer**, New York, p. 17-41, Sept. 1987

[DEUTSCH 89] DEUTSCH, P, Frameworks and reuse in the smalltalk 80 system. In: BIGGERSTAF T.; PERLIS, A. (Eds.). **Software Reusability:** Applications and Experience. New York: ACM Press, 1989.

[FAYAD 97] FAYAD, Mohamed E.; SCHIMIDT, Douglas C. Object Oriented Application Frameworks. **Communications of the ACM**, New York, v.40, n.10, Oct. 1997.

[GAMMA 94] GAMMA E. et al. **Design Patterns**: Micro-Architectures for Reusable Object-Oriented Design. Reading: Addison-Wesley, 1994.

[GOLDBERG 89] GOLDBERG, A. **Smalltalk-80:** The Language. Reading: Addison-Wesley, 1989.

[HELM 90] HELM R. et al. Contracts: Specifying Behavioral Compositions in Object Oriented Systems. In: CONFERENCE ON OBJECT-ORIENTED PROGRAMMING LANGUAGES AND APPLICATIONS, 5., 1990, Otawa, Canada. **Proceedings...** New York: ACM Press, 1990.

[JOHNSON 88] JOHNSON, R.; FOOTE, B. Designing Reusable Classes. **Journal of Object-Oriented Programming**, New York, v.1, n.12, 1988.

[JOHNSON 91] JOHNSON R. **Reusing Object-Oriented Designs.** [S.l]: University of Illinois, 1991. (Technical Report 91-1696).

[JOHNSON 92] JOHNSON, R. Documenting Frameworks Using Patterns. In: CONFERENCE ON OBJECT-ORIENTED PROGRAMMING LANGUAGES AND APPLICATIONS, 7., 1992, Vancouver, Canada, **Proceedings,** New York: ACM Press, 1992.

[JOHNSON 97] JOHNSON, R. Frameworks = Components + Patterns, **Communications of the ACM**, New York, v.40, n.10, Oct. 1997.

[KNUTH 84] KNUTH, D. Literate Programming. **Computer Journal**, New York, v.27, p. 97-111, May 1984.

[KRUEGER 92] KRUEGER, C. Software Reuse. **ACM Computing Surveys**, New York, v. 24, n.2, p.132-182, June 1992.

[LAJOIE 94] LAJOIE, R.; KELLER, R. Design and Reuse in Object-Oriented Frameworks: Patterns, Contracts, and Motifs in Concert. In: CONGRESS OF THE ASSOCIATION

CANADIENNE-FRANÇAISE POUR LA'VANCEMENT DESCIENCES, 62., 1994, Montreal, Canada. **Proceedings,** [S.1:s.n], 1994.

[LENNON 97] LENNON, J. **Hypermedia Systems and Applications.** Berlin: Spring-Verlag, 1997.

[LEVY 86] LEVY, L.S. A Metaprogramming Method and Its Economic Justification. **IEEE Transactions on Software Engineering**, New York, V.SE-12, n.2, Feb. 1986.

[MEYER 88] MEYER B. **Object Oriented Software Construction**. Hertfordshire, England: Prentice-Hall International, 1988.

[NIELSEN 90] NIELSEN, J. **Hypertext and Hypermedia**. Boston: Academic Press, 1990.

[PARCPLACE 94] PARCPLACE Inc. **VisualWorks 2.0 Reference Manual**. [S.l]: ParcPlace Inc., 1994.

[PREE 94] PREE, W. Meta-Patterns: Abstracting the Essentials of Object_Oriented Frameworks. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, 10., 1994, Bologna, Italia. **Proceedings,** Berlin: Springer-Verlag, 1994. p.150-164.

[PREE 95] PREE, W. et. al., Active Guindance Of Framework Development. **Software: Concepts and Tools**, Berlin, v.16, n. 16, p. 94-103, 1995.

[POSNAK 97] POSNAK, E.; LAVENDER; R.; VIN, H.; An Adaptative Framework For Developing Multimedia Software Components. **Communications of the ACM**, New York, v.40, n.10, Oct. 1997.

[REENSKAUG 89] REENSKAUG T.; SKARR A.L. An Environment for Literate Smalltalk Programming. In: CONFERENCE ON OBJECT-ORIENTED PROGRAMMING LANGUAGES AND APPLICATIONS, 4., 1989, **Proceedings,** New Orleans: ACM Press, 1989.

[ROBERTS 96] ROBERTS, D; JOHNSON, R. **Evolving Frameworks:** A Pattern Language for Developing Object-Oriented Frameworks. Disponível via WWW em http://st-www.cs.uiuc.edu/users/droberts/evolve.html (1996).

[RUMBAUGH 91] RUMBAUGH, J. et al. **Object Oriented Modeling and Design**. [S.l]: Prentice-Hall, 1991.

[SAMETINGER 92] SAMETINGER J.; POMPERGER G. A Hypertext System for Literate C++ Programming. **Journal of Object-Oriented Programming**, [S.l.], v.4, n.8, p. 24-29, Jan. 92.

[TALIGENT 96] TALIGENT, Inc. **Leveraging Object-Oriented Frameworks**. Disponível via WWW em http://www.taligent.com (1996).

[WIRFS-BROCKS 90] WIRFS-BROCKS R.; JOHNSON, R. Surveying Current research in Object Design. **Communications of the ACM**, New York, v.33, n.9, Sept. 1990.

[ZANCAN 99] ZANCAN, J. PRICE, R. **Um Sistema baseado em Hiperdocumentos para Apoio à instanciação de Frameworks**. Porto Alegre: CPGCC da UFRGS, 1999. Dissertação de Mestrado.